Mixed Reality

Sub-group 2
FYP Project Final Report

Members:
Ng Koon Hang (3035101554)
Wang Shi Sheng (3035085865)

Supervised by Loretta Yi-King Choi
Prepared by Ng Koon Hang (3035101554)
13 April 2017
Abstract
As information technology advances, people are trying to integrate virtual world and the reality for many reasons. As a result, mixed reality is born. For example, in design fields, people use mixed reality to preview designs so that designers can fine-tune their designs while previewing them.

In this project, we are going to present mixed reality as a multi-user interactive art gallery application. Users can place different “artworks”, such as “painting” frames, “sculptures” and 3D drawings, into the virtual world, so that when looking through the MR devices, these “artworks” and other virtual objects look as if they were integrated into the reality. Moreover, after placement, users can also interact with the devices (Tango) or by hand gestures (Hololens).

Originally this was a 4-person group project. However, after feasibility study and a small portion of single-user implementations, we think that simply making an art gallery is not a sufficient challenge for a 4-person group. As a consequence, the group was split into two, one for developing the MR art gallery, and the other group for developing an MR sports game.

This report will show our final results regarding this project with detailed explanations, and will sketch the frame of the final product. The inception phase, which is the feasibility study and SDK study, ended in October, presenting dominantly positive result. The elaboration phase, which includes the single-user implementations, is also completed. A deliverable for this part was compiled. The construction phase, which includes the core of the project: cross-platform integration between Tango and Hololens, ended in April. During the first two weeks of April, fine-tuning and testing was carried out. A final deliverable was compiled.
## Index

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>2</td>
</tr>
<tr>
<td>1 Introduction</td>
<td>4</td>
</tr>
<tr>
<td>2 Background</td>
<td>6</td>
</tr>
<tr>
<td>3 Objectives</td>
<td>8</td>
</tr>
<tr>
<td>4 Completed Phases</td>
<td>9</td>
</tr>
<tr>
<td>5 Hurdles</td>
<td>23</td>
</tr>
<tr>
<td>6 Schedule &amp; Progress</td>
<td>26</td>
</tr>
<tr>
<td>7 Conclusion</td>
<td>27</td>
</tr>
<tr>
<td>8 Acknowledgement</td>
<td>28</td>
</tr>
<tr>
<td>9 References</td>
<td>29</td>
</tr>
<tr>
<td>10 Table of Figures</td>
<td>30</td>
</tr>
</tbody>
</table>
1 Introduction
Advancing to the twenty-first century, computer technology evolved into a worldwide scale structure. One of the most important advancements in IT field is the globalization and popularization of Internet, which allows people around the world link with one another within seconds in exchange with a reasonable cost. However, the link is still not perfect -- people still unable to interact "directly" because they must still interact with one another through an interface. In order to facilitate a higher degree of integration of the virtual world and reality, different kinds of virtual reality technologies rose.

Virtual reality (VR) simulates different kinds of senses such as sight, sound and smell, making users feel as if they were actually inside the virtual world. VR has an origin in as early as the 1830s, when Charles Wheatstone stated the concept of the 3D goggles which is still in use in recent days [1]. The first commercial use of VR is argued to be in the 1920s, when VR was used to train aircraft pilots [1]. The prototype of headsets used nowadays can be dated back to the 1960s [1]. However, virtual reality, as its name suggests, merely let users experience in the virtual world, making the virtual world "real" in front of the users, which does not necessarily allow them to interact with the reality.

Augmented reality (AR) integrates virtual objects into reality in the form of showing these objects onto reality scenes without a significant delay [2], and can be classified as a part of Mixed reality (MR) [6]. The first AR system came in the 1960s, and its first use in navigation is by NASA in 1999 [3]. Now, some successful AR applications such as Google Maps by Google and Pokémon GO by Niantic become popular and global, and are sharing considerable proportions of the market. However, some applications in AR only combine virtual objects and reality in the form of "layering", that the depths of the objects in the real world are ignored thoroughly in those implementations. For example, what AR in the aforesaid popular AR game Pokémon GO does is to simply layer 2D graphics onto the scene without taking the actual environment into account, making the effects not look real enough (see Fig. 1). Google Maps, on the other hand, aligns arrows, street names and zooming
rectangles properly with the ground or the walls in the image, although this is not completely MR because the street views used are not instant updated images.

Fig. 1 The Pokéball standing still on the wall.
2 Background

As MR technology with depth perception is still waiting for further development, its support is still very limited such that specialized devices must be used to achieve such functions. Recently, global IT companies Google and Microsoft each individually released an MR system which is capable of processing depth data, namely Google Tango and Microsoft HoloLens.

Google Tango uses depth sensors to measure distances [4]. There are several implementations of such sensors, which depend on the makers of the devices implementing Tango [4]. Google Tango also uses the camera to capture the reality scene and display on the screen, which forms a base for other virtual objects to be placed on (see Fig. 2).

Microsoft HoloLens also use sensors [5]. Users look through a goggle-like device with a pair of transparent eyepieces. The images are projected onto the eyepieces, so looking through the eyepieces allows users to look both the reality scene and the virtual objects at the same time. Comparing to Google Tango, Microsoft HoloLens allows users to
use gestures to manipulate virtual objects, and gives users more real experience (see Fig. 3)[5].

**Fig. 3** Introduction to applications of Microsoft HoloLens [5]. Captured from the website on October 22, 2016.
3 Objectives

In this project, we are going to construct an MR art gallery application which can achieve the following goals:

1. The application features a multi-user scene, that is, multiple users can manipulate the world simultaneously, and all manipulations shall show instantaneously.

2. Users using Google Tango-supporting devices and Microsoft HoloLens can look directly through the devices with virtual objects located suitably in the virtual world, and place virtual objects, such as virtual furniture into the world.

3. Users, no matter what platform they use, shall be able to locate preset sculptures, which are chosen from a library of models, on the ground. Users using Google Tango-supporting devices can also place photo frames with pictures or comments on the wall. Users using Microsoft Hololens can also draw 3D drawings using hand gestures within the virtual space.

The application should be able to allow at least 4 people to simultaneous participate in the construction, and after one device finishes an operation, other connected devices should respond within a latency of less than 1 second. Obstacles within the space, if any, shall be detected and “artworks” behind shall become invisible to the users.
4 Completed Phases

1. Feasibility Study
The feasibility study of the project plan was finished in the third week of October. The project plan was evaluated feasible unanimously by all group members, with some exceptions which will be explained in the Results part.

Methodology
The feasibility study was done individually, one for each MR system. The first half of the development routine will follow this sub-group division. Both sub-groups carried out API research on the systems as preparation for subsequent development.

Ng, who is responsible for Google Tango, downloaded some recommended applications exclusively for Project Tango from Google Play, and conducted some test runs on each of the applications. Besides downloading demo applications, Ng also compiled a demo "Tango Test" following Tango Unity SDK reference (which can be retrieved from https://developers.google.com/tango/apis/unity/).

Applications used in this stage include: (1) Measure / Measure It (as both share similar functionalities, they are grouped as one bundle of applications), (2) Tango Constructor, and (3) Solar Simulator for Dev Kit.

Tests for (1) were designed to study on the characteristics and constraints of the camera with depth perception. We put the focus to a specific point on a wall from different distances, to points on different surface materials, in different lightings, and in different planes.

Tests for (2) were designed to study the ability of construction of 3D models from an amount of points, which Tango system returns from the raw data captured by the camera. We tested the application against different real objects to see whether the objects
are reconstructed in the application properly.

Tests for (3) were designed to study on positioning of virtual objects according to the parameters from the reality environment. For the given two points (Sun and Neptune respectively), we check whether the planets in between are properly placed by the application. This test is relatively simple comparing to the other two applications because the functions of this application are limited.

Results (Tango)

i. Measure / Measure It

The camera has a probability of failing to detect a distance of less than 0.4 meters or larger than 5 meters, and detection usually fails in dim light. Solid surfaces that were difficult to change their shapes were easier to detect.

The application was able to recognize a plane whose all points are on different planes (see Fig. 4).

![Four triangles detected by Measure It, whose vertices are on at least 2 different planes.](image)

Fig. 4 Four triangles detected by Measure It, whose vertices are on at least 2 different planes.

ii. Tango Constructor

Fig. 5 shows that 3D construction is possible, but the precision is not enough to remodel the target chair. The chair in Fig. 5(b) shows four legs, each 2 centimeters in diameter, but they were not promptly captured by the constructor as in Fig. 5(a). It is not plausible that the meshes of the virtual
furniture are constructed this way. Consequently, the original proposal that we included the functions to extract a 3D model from real furniture is turned down. However, despite the fact that the model reconstructed is rough, the model should be sufficient to serve as an obstacle which can block the objects behind.

Fig. 5 (a) The model of a chair constructed by Tango Constructor, and (b) the original chair.

iii. Solar Simulator for Dev Kit
The planets were properly placed across the space for the given two points. This is not a remarkable result, because Measure It has shown this by stretching a dotted line between two points across the space between. However, it also shows the basic enlargement and diminishment process of a system in a different scale, and thus it is possible to calibrate a linear world by two points, interpolating and extrapolating the points between and beyond the two fixed points.

iv. Demo application as instructed by Tango Unity SDK
Most basic functions, such as camera orientation and trace tracking, can be implemented without extra lines of code, because the prefab provided already carries the corresponding scripts which will be called at runtime. For advanced functions, the SDK provides a detailed documentation as a reference. As an SDK, Tango Unity SDK makes implementation easier for developers, especially who are new to Tango.

However, because there is possibly a bug in the SDK itself,
the demo applications crashed within one minute (see "Hurdles"). Fortunately, the problem was resolved by disabling two of the SDK settings before publishing the application. Since then, the demo applications has not crashed.

2. Implementation in Single User Environment
   In order to improve feasibility, we started the implementation of proposed functions right after feasibility study. This phase is divided into the following parts: (1) placement of objects, (2) 3D drawing, and (3) object manipulation. At the time when the report is prepared, (1) and (3) is completed for Tango, and (2) and (3) is completed for Hololens.

Completed Parts (Tango)
   i. Placement of Objects
      In the proposed plan, as stated in the Objective section, there will be a function for users using Google Tango-supported devices or Microsoft HoloLens, which is to place objects on the floor or on the wall of the space. For Google Tango-supported devices, an exclusive function, which is to hang up photo frames with images and comments on the walls of the room, is available.

      One of the Tango demo applications "Measure It" is already able to recognize areas of different shapes, and to show the bounded area in yellow. This function is actually derived from this discovery during feasibility study, with alterations which make this function more related to the original project plan, and more applicable to real life.

      The deliverable for Tango at this stage, however, only includes placement of painting frames on the wall, as the models of the sculptures are still in progress as of the preparation of this report.

Methodology
The model of the frame is made that the origin of the local coordinate system is at the back of the center of the frame. This allows the frame to stick on the wall without hiding any of the front part. This also agrees with the real case, when we hang up a photo frame none of the parts should submerge into the wall.

First, the user locates a rectangular region, whose aspect ratio is the same of the source picture. As the selected region must be rectangular and has the same aspect ratio as the source picture, the positioning algorithm must cater these constraints. The algorithm of placing a picture is simplified, although an assumption that the camera is calibrated properly is considered. Below we will depict how this implementation actually works.

i. User chooses a point on the wall (see Fig. 6(a)) in "Picture Mode". This will start the placement process. The application will make API calls to retrieve the corresponding world coordinates with respect to the 2D coordinates the user selected. This point will be the center point of the picture frame.

ii. User will be prompted to select a point on the right of the first point (see Fig. 6(b)). This is for the calculation of the normal, because the plane finding seems to be flawed as of the preparation of the report (see “Hurdles”). If plane finding of Tango works well, this step will be skipped.

iii. After selecting the second point, a user interface will appear to let the user choose a picture from the device, and user fills in the title, source and the description (optional) of the picture. User also assigns the length of the longer side of the picture. This length will be with respect to the real world.

iv. Let the vector from the first point to the second point be \( \mathbf{u} \), and the vector pointing up and parallel to the wall be \( \mathbf{v} \). The normal \( \mathbf{n} \) will be calculated by \( \mathbf{n} = \mathbf{u} \times \mathbf{v} \).
v. By adjusting the Euler angles, position, and setting the picture of the photo frame instantiated, the photo frame is hung on the wall properly (see Fig. 6(d)). The frame is 7.5 cm x 5 cm thick and the picture is 2.5 cm inside the frame.

vi. If user chooses a picture, the application will call Physics.Raycast to find a picture on that point. The picture frame will then be returned and a tag, which is aligned with the picture and the real world, will appear, showing the information of the picture, and will disappear automatically after three seconds (see Fig. 6(e), (f)). Multiple tags can appear at the same time, if the time interval between multiple selections are shorter than the display time of the tags. The tags will be generated in different size if the picture is investigated from different distances.

The prefabs are made that the origin of the local coordinate system is at the center of the base of the prefab, which faces down. This disallows the object to submerge into the ground, and renders the subsequent transformations clearer. This also agrees with the real case, in which no part of a sculpture should submerge into the solid ground.

If the coordinate system is calibrated and the application is started at upright position, only the rotation obtained from calibration will be needed to place the prefab onto the floor. Therefore, the algorithm is rather straightforward and fundamental. The implementation works as follows:

i. User chooses a point on the floor (see Fig. 6(g)) in "Sculpture Mode". This will start the placement process. The application will make API calls to retrieve the corresponding world coordinates with respect to the 2D coordinates the user selected. This point will be the center point of the prefab.

ii. A user interface will appear to let the user choose a
prefab from a predetermined list of prefabs and enter the size of the object (see Fig. 6(h)).

iii. The prefab will then be placed on the indicated position with the base as pivot point, with the identity rotation with respect to the calibrated coordinate system (see Fig. 6(i)).

iv. If the sculpture is selected by the user, the sculpture can then be manipulated by the user (see the next section).
b. Manipulations of Objects

Despite the fact that support of gesture on Tango is considerably less than on Hololens, that only touch gestures are supported rather than hand gestures which Hololens supports, some of the basic object manipulations are still feasible by interactions on the Tango devices, such as displacing and scaling. These two manipulations are actually what we have implemented on Tango, while on Hololens rotation is also available.

**Impossibility on implementing object rotation on Tango**

According to (a), an object is defined with the local origin, which is referenced by transformations in Unity, to be the center of the base of the object. While constructing the prefabs, the structure is placed inside a parent GameObject object, and the transformations is therefore applied to this parent as a whole. Moreover, the objects are asserted to be placed on the floor with the base facing down.

As Tango is a tablet or smart phone, it is important to realize that the control region is 2-dimensional, comparing to the 3-dimensional control region of Hololens. As the objects are asserted to have their bases facing down, the only rotation available is along Y-axis (corresponding to Z-axis in reality). However, rotating along the Y-axis requires 3-dimensional hand gesture control, since the gesture of the rotation involves depth. As a consequence, object rotation that keeps the base facing down is impossible on Tango.
Methodology
Both algorithms are as straightforward as the one which places a sculpture because of the lower difficulty on retrieving input coordinates.

For displacing a sculpture, the algorithm is as follows:

i. User selects a sculpture. Using Physics.Raycast together with the API call that retrieves the corresponding point in the reality, the selected sculpture will be assigned to a variable call "controlling" and will be locked so that other users cannot control it at the moment.

ii. User displaces the sculpture by moving the device around and/or moving the finger on the tablet. The new position will be calculated using API calls per frame, and is calculated so that even if there is no corresponding point due to out of range detection the sculpture will not be out of control.

iii. User releases the finger to locate the sculpture. This also releases the lock of the sculpture.

For scaling a sculpture, the algorithm starts at (i) of the previous algorithm, and is continued as follows:
i. Users places the second finger at a distance from the first finger. The distance will be calculated and stored.

ii. Users moves the fingers. The new distance will be calculated per frame. Let $d_a$ be the distance when the second finger has just touched the tablet, and $d_b$ be the instantaneous distance of the fingers, $s_a$ and $s_b$ be the initial and final scale of the sculpture. Then, $s_b = s_a \times \frac{d_b}{d_a}$. The instantaneous scale will be reflected per frame.

iii. Users releases the second finger. The calculation will stop.

iv. The algorithm continues at (iii) of the previous algorithm.

3. Implementation in Multi User Environment

In the proposed plan, as stated in the Objective section, there will be a linkage between Tango-supported devices or Microsoft HoloLens, which synchronizes the game objects in a session. The implementation is achieved through Photon Unity Network, which is a free cloud game network providing instant cloud network calculation. Photon automatically optimizes the traffic control, which "guarantee low latency and shortest round-trip times for your players worldwide" [8:1].

However, due to differences in rendering efficiency between Tango and Hololens, some game objects are not synchronized between two platforms (see Hurdles).

a. Connection

The connection is achieved by two parts: Photon Unity Network, which handles state updates, and an simple image server, which stores images uploaded through creation of picture frames.

**Photon Unity Network (PUN)**

i. Protocol

The protocol separates into three parts.
A. A BasicObject class provides the basis of the parts. It consists of the following components: object ID, object name, transform, creator ID and hash. It also provides a serialize and a de-serialize interface, which can be extended to process the data for sending out or for display.

B. A PrefabObject class, which is extended from BasicObject class, stores the data of a sculpture. A Boolean variable "controlledBySb" is used to record whether the sculpture is being controlled by someone else. If so, if the user attempts to control the object, a warning message will show in the bottom left corner.

C. A FrameObject class, which is also extended from BasicObject class, stores the data of a picture frame. Three fields are added: "pictureID" which records the ID of the picture, which is obtained when the picture is uploaded to the image server, "longLength" which records the longer length of the picture in meters, and "pictureData" which records the title, source and description of the picture. The picture data is also synchronized to ensure that other users can see the description of the picture when selected the picture.

ii. Passage of Data

The passage of data is administered in the rate of approximately 10 FPS. In each pass, each game object on the scene either serializes and send the data of an object to the server, or receives, de-serializes the data from the server and alters the game object. Before sending data to the server, the transform of the game object will be converted to local coordinates first (see 4.3(b) Coordinate Calibration for details). Similarly, after receiving data from the server, the transform of the game object will be converted back to global coordinates for proper placement.

Image Server

The implementation of the image server consists of a PHP
server, which only contains a single PHP page, which receives the raw bytes of the image and writes into the server as an image file. The interaction between the application and the image server is as follows:

i. After selecting an image from the local storage during the creation of a picture frame, the image will be converted to a Base64 string, and sent to the image server.

ii. The script will convert back the Base64 string to raw bytes, and check the signature of the image to prevent malicious images. The script checks whether the picture is in JPG or PNG format. If so, the MD5 hash of the picture will then be calculated, and the image will be saved into the storage in the server using the hash as the file name.

iii. The file name, if applicable, is then returned with a status in the form of JSON back to the application (see Fig. 6(d)). The application then creates the image frame by passing the returned file name as picture ID. Then, by calling PhotonNetwork.Instantiate, the picture ID will also be sent to the network for other users.

iv. When other devices receives this new picture frame, the photo frame will be instantiated, and at the same time a WWW request is passed to the image server to download the corresponding image. Upon completion of the download, the texture of the "Quad" in the picture frame prefab will be changed to the download texture, completing the initialization, and mark the picture as complete to prevent subsequent redundant downloads. This is because once the image is set no more changes will be allowed. If the download fails, the download process will start again in the next synchronization cycle.

b. Coordinate Calibration

While the application in Hololens uses Vuforia, which is an AR marker recognition service, the application in Tango continues to use functions in Tango because of the
incompatibility with Vuforia (see Hurdles). As a result, this Tango application uses manual calibration, which needs two markers to finish. The algorithm is as follows:

i. User selects the center of the marker. The marker will be the same as that used in Hololens. The world coordinates of the point will be recorded as P1.

ii. User selects the upper center of the marker, or an extra small marker beyond the upper center of the marker. The world coordinates of the point will be recorded as P2.

iii. The normalized vector of the difference of the two points \( P_2 \) will be calculated as \( P_z = \frac{P_2 - P_1}{||P_2 - P_1||} \).

iv. P1 will be the calibration offset, and the difference R of the angle between the vector and the unit vector of Z-axis will be the calibration rotation. Now, the Z-axis is calibrated.

v. User selects the right center of the marker, or any place on the right on the same plane. The world coordinates of the point will be recorded as P3.

vi. The normalized vector of the difference of the two points \( P_3 \) will be calculated as \( P_x = R \frac{P_3 - P_1}{||P_3 - P_1||} R^{-1} \).

vii. Rotate the calibration rotation along Z-axis by \( \sin^{-1} P_x \) radians counterclockwise to complete the calibration.

The conversion between world transformation and calibrated transformation are as follows:

Let P be the calibration offset, R be the calibration rotation, P0 be the world position, R0 be the world rotation, P1 be the calibrated position, R1 be the calibrated rotation. Then,

\[
P_0 = R P_1 R^{-1} + P \quad \text{(1)}
\]
\[
R_0 = R_1 R \quad \text{(2)}
\]
\[
P_1 = R^{-1} (P_0 - P) R \quad \text{(3)}
\]
\[
R_1 = R_0 R^{-1} \quad \text{(4)}
\]
c. **Adaptation of Multi-player functions on Single-player implementations**
Some single-player functions are amended to adapt the multi-player purposes.

i. All Instantiate function on creating game objects (except the UI instantiation) are switched to PhotonNetwork.Instantiate so that the game objects are instantiate through PUN, making other devices able to see the game objects.

ii. Time needed to place a picture frame becomes longer because the image needs to be uploaded to the image server (see 4.3(a) on Image Server).

4. **Testing and Debugging**
Testing shows that no crashes are caused by our code. However, the Tango device itself sometimes crashes at startup stating that "Google 書報攤 has crashed". We doubt that the testing device itself is unstable.
5 Hurdles

Tango
During the feasibility study and single-user implementation, we encountered some hurdles that slowed down our progress by different levels. Some of the hurdles are already resolved, while we are still finding the solutions to some of those difficulties.

We encountered the first hurdle in the feasibility study stage. When we completed compiling the first Tango demo application, the application crashed at random times, and the crash times after the launches of the application were record from a few seconds to several minutes. As the ADB was not able to detect the device we are using to develop the application, we had no data about the error log generated by the system. This is a severe issue because this can directly prevent us from testing further if the issue cannot be resolved during development. Fortunately, after some researches on the specific issue, the main reason of the crash was finally revealed to be the bug in the Tango Unity SDK itself. This bug causes Tango applications to crash if two of the options in the settings of the Tango Unity SDK is enabled [7]. We tried this solution and the problem was resolved. Further study found that the reason ADB was not able to detect the device was that the driver was not properly installed despite being downloaded using SDK Manager.

The second hurdle is in the world coordinate synchronization. As the application needs to show the same world regardless of what device the user is using, this synchronization is essential. If synchronization is not carried out before the user explores the "world", or if the synchronization makes an error, the "world" will show distorted and dislocated on the user's device. As a result, if any calibration error arises, it will become a major issue to us. However, even in the methodology part we stated that it is relatively easy to implement the calibration, deriving such an algorithm is actually difficult. Unity uses Euler angles to represent the rotation of the camera, so the transform matrix is not so clear. Moreover, the calculations are actually complex, and implementing these calculations seems to require a certain amount of knowledge in computer vision.
One of the solutions, and which was the solution implemented in single-user implementation as a test, is to instruct users to start the application while posing their devices in a specific manner at a specific position. This solves the synchronization problem because all sessions start with the same initial state, but this greatly limits the flexibility, such that all sessions must start alike, thus the application will become unable to adapt to the environments of different users freely.

We tried to solve this by using Vuforia, which can automatically detect a marker, returning the transform of the marker with respect to the camera. This works on Hololens, but not on Tango, as it makes the application crashes on startup on Android 4.4.2. As a reference, we made a Vuforia test application, which solely contains the Vuforia plugin, but not the Tango plugin. The test application crashed on Tango as well, but it did not crash on Android 7.0. After some further study, we found that in a discussion about this issue there is a comment saying there may be incompatibility between two AR systems, that "both libraries trying to access the camera concurrently" [9:#6]. However, a further ADB debug log inspection shows that the crash is caused by a "NoSuchMethodError" referencing the method "Ljava/nio/channels/ReadableByteChannel;.close()V". It seems to be SDK issues so this method is not possible on Tango.

After some further research on Quaternion, we found that quaternion computation makes the calibration easier, and we finally succeeded on deriving the equation using quaternion computations.

The third hurdle is the automatic plane finding for Tango. When processing the data from the point cloud, although the function for getting nearest point for a given 2D point works normally, the function for getting the plane do not work. The function provided by the SDK only returns a bunch of zeros or null values. Without this function, two points if one vector is presumed, or three points in general case, are needed to identify a plane. It is possible to randomly choose three points from the point cloud automatically for calculation, but errors may arise if there are multiple surfaces in the scene. One solution is to find two more neighbouring points around the nearest point returned by the SDK, but there may be a computational cost trade-off.
The fourth hurdle is that Tango is more computationally inefficient than Hololens. 3D drawings and 3D reconstructions were first included into the project. However, after some researches, we found that even a simple segment of 3D drawing produces triangles in the order of $10^3$, while 3D reconstructions produces triangles in the order of $10^4$ per prefab. Although Hololens is able to handle this amount of renderings, because Tango is slower comparing to Hololens, we concluded that the same rendering has a potential to cause Tango to freeze or to crash. As a result, in order to prevent such potential hazards, the synchronization ignores 3D drawings and 3D reconstructions on Hololens.
## 6 Project Schedule & Overall Progress

<table>
<thead>
<tr>
<th>Task</th>
<th>Progress</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feasibility study</td>
<td>Completed October 2016</td>
</tr>
<tr>
<td>Single-user implementation</td>
<td>Completed December 2016</td>
</tr>
<tr>
<td>Multi-user networking</td>
<td>Completed March 2017</td>
</tr>
<tr>
<td>Merger of systems</td>
<td>Completed April 2017</td>
</tr>
<tr>
<td>Testing</td>
<td>Completed April 2017</td>
</tr>
</tbody>
</table>
7 Conclusion
We have completed the project, and have tested that the functionalities work as expected.

The feasibility study was finished as expected in October, which gave us that the project plan is basically feasible in wide sense. After the tests of four demo applications, we conclude that most of our planned functionalities are feasible to implement on Tango and Hololens altogether, and the implementation is easier than we expected before. This is a positive signal for us to continue to work on this project.

The single-user implementation was finished as expected in December. At the time of December, except the coordinate system calibration, which further study was planned in the next phase, all of the other implementations, which are the implementations of the basic functionalities, were fundamentally completed. Although some optimizations were still needed, they were functional, and excluding the deliverables from the feasibility study, 2 deliverables for these implementations were already available for testing.

The server systems were finished in early March. We implemented a Photon Unity Network and a PHP server, which have different roles in this project. Connections between servers and the application work properly.

Merger of systems, together with the coordinate calibration, were completed in early April. Although two systems uses different calibration method, the result are expected to be the same among the systems. However, some parts were concluded infeasible to merge (see Hurdles).

Functionality test was finished at the time the report is prepared. We concluded that the functionalities works as expected.
8 Acknowledgement

We hereby acknowledge Prof. Loretta Yi-King Choi for supervising and consulting us to proceed on the project.

We hereby acknowledge School of Computer Science, Faculty of Engineering, HKU for providing necessities for this project.
9 References


# Table of Figures

<table>
<thead>
<tr>
<th>Fig. 1</th>
<th>The Pokéball standing still on the wall.</th>
<th>P. 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fig. 2</td>
<td>Using a Google Tango-supported tablet to measure distances with an MR application &quot;Measure&quot;.</td>
<td>P. 6</td>
</tr>
<tr>
<td>Fig. 3</td>
<td>Introduction to applications of Microsoft HoloLens [5]. Captured from the website on October 22, 2016.</td>
<td>P. 7</td>
</tr>
<tr>
<td>Fig. 4</td>
<td>Four triangles detected by Measure It, whose vertices are on at least 2 different planes.</td>
<td>P. 10</td>
</tr>
<tr>
<td>Fig. 5</td>
<td>(a) The model of a chair constructed by Tango Constructor, and (b) the original chair.</td>
<td>P. 11</td>
</tr>
<tr>
<td>Fig. 6</td>
<td>Workflow of the Tango deliverable, demonstrating the implementation of placements and the runtime result.</td>
<td>P. 15-16</td>
</tr>
<tr>
<td>Fig. 7</td>
<td>Illustration showing how depth is necessary for a local Y-axis rotation.</td>
<td>P. 17</td>
</tr>
</tbody>
</table>